Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method for preserving data constraints during parallel ~~apply in~~ application of asynchronous replication transactions ~~replication~~ in a database system, the method comprising:

(a) receiving and examining a transaction message comprising one or more row changes;

(b) determining ~~if~~ whether at least one row change in the transaction message ~~has~~ is affected by a data ~~constraints~~ constraint;

(c) ~~if so~~ responsive to at least one row change in the transaction message being affected by a data constraint, determining ~~if~~ whether there is a constraint violation between the at least one row change in the transaction message ~~has a constraint violation with~~ and a row change in at least one preceding non-completed transaction message; ~~and~~

(d) ~~if so~~ responsive to there being a constraint violation between the at least one row change in the transaction message and the row change in the at least one preceding non-completed transaction message, holding the transaction message until application of the at least one preceding non-completed transaction message ~~completes~~ is completed;

(e) responsive to application of the at least one preceding non-completed transaction message being completed, placing the transaction message on a work queue; and

(f) subsequent to placing the transaction message on the work queue, applying the transaction message in parallel with one or more other transaction messages on the work queue.

2. (Original) The method of claim 1, wherein the determining (b) comprises:

(b1) determining that the at least one row change in the transaction message has secondary unique constraint; and

(b2) recording column values for the secondary unique constraints of the at least one row change in the transaction message, when the at least one row change is a row insert or update.

3. (Original) The method of claim 2, wherein if the column value for the secondary unique constraint is not known, then record an "unknown" value.

4. (Original) The method of claim 2, wherein the determining (c) comprises:

(c1) comparing the column values for secondary unique constraints for the at least one row change in the transaction message with recorded column values for secondary unique constraints for the row change in the at least one preceding non-completed transaction message.

5. (Original) The method of claim 4, wherein the holding (d) comprises:

(d1) determining that column values for secondary unique constraints for the at least one row change in the transaction message matches recorded column values for secondary unique constraints for the row change in the at least one preceding non-completed transaction message; and

(d2) holding the transaction message until the application of the at least one preceding non-completed transaction message completes.

6. (Original) The method of claim 5, wherein the holding (d) further comprises:

(d3) determining that the column values for the secondary unique constraints for the at least one row change in the transaction message do not match the recorded column values for the secondary unique constraints for the row change in the at least one preceding non-completed transaction message; and

(d4) placing the transaction message on the work queue to be applied in parallel with the other transaction messages on the work queue.

7. (Original) The method of claim 6, wherein the placing (d4) comprises:

(d4i) applying the transaction message to a target table;

(d4ii) determining if a constraint violation results from the application of the at least one row change in the transaction message; and

(d4iii) periodically retrying the application of the at least one row change in the transaction message, if the constraint violation results.

8. (Original) The method of claim 7, wherein the periodically retrying (d4iii) comprises:

(d4iiiA) comparing a next transaction message on the work queue with the transaction message to be retried;

(d4iiiB) determining if the next transaction message is older than the transaction message to be retried;

(d4iiiC) placing the transaction message to be retried back on the work queue, if the next transaction message is older; and

(d4iiiD) applying the transaction message to be retried, if the next transaction message is not older.

9. (Original) The method of claim 1, wherein the determining (b) comprises:

(b1) determining that the transaction message has a referential integrity constraint.

10. (Original) The method of claim 9, wherein the determining (c) comprises:

(c1) determining that a target table is a parent table of the referential integrity constraint; and

(c2) determining if a row operation of the transaction message is a row insert type; and

(c3) determining if the subsequent transaction message to a child table is the row insert type, if the row operation of the transaction message is the row insert type.

11. (Original) The method of claim 10, wherein the holding (d) comprises:

(d1) holding the subsequent transaction message to the child table until the transaction message completes, if the row operation of the transaction message is not the row insert type;

(d2) holding the subsequent transaction message to the child table, if the subsequent transaction message to the child table is not the row insert type and the row operation of the transaction message is the row insert type; and

(d3) placing the subsequent transaction message to the child table on the work queue to be applied in parallel with the transaction message, if the subsequent transaction message to the

child table is the row insert type and the row operation of the transaction message is the row insert type.

12. (Original) The method of claim 11, wherein the placing (d3) comprises:

(d3i) applying the subsequent transaction message to the child table;

(d3ii) determining if a constraint violation results from the application of the subsequent transaction message; and

(d3iii) periodically retrying the application of the subsequent transaction message, if the constraint violation results.

13. (Original) The method of claim 12, wherein the periodically retrying (d3iii) comprises:

(d3iiiA) comparing a next transaction message on the work queue with the transaction message to be retried;

(d3iiiB) determining if the next transaction message is older than the transaction message to be retried;

(d3iiiC) placing the transaction message to be retried back on the work queue, if the next transaction message is older; and

(d3iiiD) applying the transaction message to be retried, if the next transaction message is not older.

14. (Original) The method of claim 9, wherein the determining (c) comprises:

(c1) determining that a target table is a child table of the referential integrity constraint; and

(c2) determining if a row operation of the transaction message is a row delete type; and

(c3) determining if the subsequent transaction message to a parent table is the row delete type, if the row operation of the transaction message is the row delete type.

15. (Original) The method of claim 14, wherein the holding (d) comprises:

(d1) holding the subsequent transaction message to the parent table until the transaction message completes, if the row operation of the transaction message is not the row delete type;

(d2) holding the subsequent transaction message to the parent table, if the subsequent transaction message to the parent table is not the row delete type and the row operation of the transaction message is the row delete type; and

(d3) placing the subsequent transaction message to the parent table on the work queue to be applied in parallel with the transaction message, if the subsequent transaction message to the parent table is the row delete type and the row operation of the transaction message is the row delete type.

16. (Original) The method of claim 15, wherein the placing (d3) further comprises:

(d3i) applying the subsequent transaction message to the parent table;

(d3ii) determining if a constraint violation results from the application of the subsequent transaction message; and

(d3iii) periodically retrying the application of the subsequent transaction message, if the constraint violation results.

17. (Original) The method of claim 16, wherein the periodically retrying (d3iii) comprises:

(d3iiiA) comparing a next transaction message on the work queue with the transaction message to be retried;

(d3iiiB) determining if the next transaction message is older than the transaction message to be retried;

(d3iiiC) placing the transaction message to be retried back on the work queue, if the next transaction message is older; and

(d3iiiD) applying the transaction message to be retried, if the next transaction message is not older.

18. (Original) A method for preserving data constraints during parallel apply in asynchronous transaction replication in a database system, comprising:

(a) identifying a transaction message as a cascade delete;

(b) determining that a source of the transaction message is not a leaf table;

(c) placing each subscription for the transaction message onto a stack and placing row operations for each subscription into a reorder list, wherein the subscriptions are placed onto the stack in order of execution, wherein the row operations are placed into the reorder list in the order of execution; and

(d) adding the row operations for each subscription in the stack back to the transaction message, wherein the row operations are added in a reverse order of execution, wherein the subscriptions are added in the reverse order of execution.

19. (Original) The method of claim 18, further comprising:

(e) sending the transaction message to be applied to a target table.

20. (Original) The method of claim 18, further comprising:

(e) applying the transaction message at a target table.

21. (Original) A method for preserving data constraints during parallel apply in asynchronous transaction replication in a database system, comprising:

(a) receiving a message to perform an initial load of a target table;

(b) determining that the target table is a child table of referential integrity constraints;

(c) saving the referential integrity constraints for the target table;

(d) dropping the referential integrity constraints from the target table;

(e) loading the target table in parallel with a loading of a parent table of the referential integrity constraints;

(f) begin applying change data to the target table once loading is done;

(g) waiting for the parent table to finish loading, if the parent table has not yet finished loading; and

(h) adding the referential integrity constraints back into the target table.

22. (Original) The method of claim 21, wherein the adding (h) comprises:

(h1) for each child referential integrity constraint for the target table, determining if a parent schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraints back into a child table of the target table, if the parent schema and table name of the referential constraint do not match the target table name in the subscription.

23. (Original) The method of claim 22, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the parent schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the child table, if the state of the subscription is active or inactive.

24. (Original) The method of claim 21, wherein the adding (h) comprises:

(h1) for each parent referential integrity constraint for the target table, determining if a child schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraint back into the target table, if the child schema and table name of the referential constraint do not match the target table name in the subscription.

25. (Original) The method of claim 24, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the child schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the target table, if the state of the subscription is active or inactive.

26. (Original) A method for preserving data constraints during parallel apply in asynchronous transaction replication in a database system, comprising:

(a) receiving a message to perform an initial load of a target table;

(b) determining that the target table is a parent table of referential integrity constraints;

(c) saving the referential integrity constraints for a child table of the target table;

(d) dropping the referential integrity constraints from the child table;

(e) loading the target table in parallel with a loading of the child table;

(f) begin applying change data to the target table once loading is done;

(g) waiting for the child table to finish loading, if the child table has not yet finished loading; and

(h) adding the referential integrity constraints back into the child table.

27. (Original) The method of claim 26, wherein the adding (h) comprises:

(h1) for each child referential integrity constraint for the target table, determining if a parent schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraints back into the child table, if the parent schema and table name of the referential constraint do not match the target table name in the subscription.

28. (Original) The method of claim 27, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the parent schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the child table, if the state of the subscription is active or inactive.

29. (Original) The method of claim 26, wherein the adding (h) comprises:

(h1) for each parent referential integrity constraint for the target table, determining if a child schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraint back into the target table, if the child schema and table name of the referential constraint do not match the target table name in the subscription.

30. (Original) The method of claim 29, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the child schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the target table, if the state of the subscription is active or inactive.

31. (Currently Amended) A computer readable medium ~~with~~ encoded with a computer program ~~instructions~~ for preserving data constraints during parallel ~~apply in~~ application of asynchronous replication transactions ~~replication~~ in a database system, the computer program comprising instructions for:

(a) receiving and examining a transaction message comprising one or more row changes;

(b) determining ~~if~~ whether at least one row change in the transaction message ~~has~~ is affected by a data ~~constraints~~ constraint;

(c) ~~if so~~ responsive to at least one row change in the transaction message being affected by a data constraint, determining ~~if~~ whether there is a constraint violation between the at least one row change in the transaction message ~~has a constraint violation with~~ and a row change in at least one preceding non-completed transaction message; ~~and~~

(d) ~~if so~~ responsive to there being a constraint violation between the at least one row change in the transaction message and the row change in the at least one preceding non-completed transaction message, holding the transaction message until application of the at least one preceding non-completed transaction message ~~completes~~ is completed;

(e) responsive to application of the at least one preceding non-completed transaction message being completed, placing the transaction message on a work queue; and

(f) subsequent to placing the transaction message on the work queue, applying the transaction message in parallel with one or more other transaction messages on the work queue.

32. (Original) The medium of claim 31, wherein the determining (b) comprises:

(b1) determining that the at least one row change in the transaction message has secondary unique constraint; and

(b2) recording column values for the secondary unique constraints of the at least one row change in the transaction message when the at least one row change is a row insert or update.

33. (Original) The medium of claim 32, wherein if the column value for the secondary unique constraint is not known, then record an "unknown" value.

34. (Original) The medium of claim 32, wherein the determining (c) comprises:

(c1) comparing the column values for secondary unique constraints for the at least one row change in the transaction message with recorded column values for secondary unique constraints for the row change in the at least one preceding non-completed transaction message.

35. (Original) The medium of claim 34, wherein the holding (d) comprises:

(d1) determining that column values for secondary unique constraints for the at least one row change in the transaction message matches recorded column values for secondary unique constraints for the row change in the at least one preceding non-completed transaction message; and

(d2) holding the transaction message until the application of the at least one preceding non-completed transaction message completes.

36. (Original) The method of claim 35, wherein the holding (d) further comprises:

(d3) determining that that column values for the secondary unique constraints for the at least one row change in the transaction message do not match the recorded column values for the secondary unique constraints for the row change in the at least one preceding non-completed transaction message; and

(d4) placing the transaction message on the work queue to be applied in parallel with the other transaction messages on the work queue.

37. (Original) The medium of claim 35, wherein the placing (d4) comprises:

(d4i) applying the transaction message to a target table;

(d4ii) determining if a constraint violation results from the application of the at least one row change in the transaction message; and

(d4iii) periodically retrying the application of the at least one row change in the transaction message, if the constraint violation results.

38. (Original) The medium of claim 37, wherein the periodically retrying (d4iii) comprises:

(d4iiiA) comparing a next transaction message on the work queue with the transaction message to be retried;

(d4iiiB) determining if the next transaction message is older than the transaction to be retried;

(d4iiiC) placing the transaction message to be retried back on the work queue, if the next transaction message is older; and

(d4iiiD) applying the transaction message to be retried, if the next transaction message is not older.

39. (Original) The medium of claim 31, wherein the determining (b) comprises:

(b1) determining that the transaction message has a referential integrity constraint.

40. (Original) The medium of claim 39, wherein the determining (c) comprises:

(c1) determining that a target table is a parent table of the referential integrity constraint; and

(c2) determining if a row operation of the transaction message is a row insert type; and

(c3) determining if the subsequent transaction message to a child table is the row insert type, if the row operation of the transaction message is the row insert type.

41. (Original) The medium of claim 40, wherein the holding (d) comprises:

(d1) holding the subsequent transaction message to the child table until the transaction message completes, if the row operation of the transaction message is not the row insert type;

(d2) holding the subsequent transaction message to the child table, if the subsequent transaction message to the child table is not the row insert type and the row operation of the transaction message is the row insert type; and

(d3) placing the subsequent transaction message to the child table on the work queue to be applied in parallel with the transaction message, if the subsequent transaction message to the child table is the row insert type and the row operation of the transaction message is the row insert type.

42. (Original) The medium of claim 41, wherein the placing (d3) comprises:

(d3i) applying the subsequent transaction message to the child table;

(d3ii) determining if a constraint violation results from the application of the subsequent transaction message; and

(d3iii) periodically retrying the application of the subsequent transaction message, if the constraint violation results.

43. (Original) The medium of claim 42, wherein the periodically retrying (d3iii) comprises:

(d3iiiA) comparing a next transaction message on the work queue with the transaction message to be retried;

(d3iiiB) determining if the next transaction message is older than the transaction message to be retried;

(d3iiiC) placing the transaction message to be retried back on the work queue, if the next transaction message is older; and

(d3iiiD) applying the transaction message to be retried, if the next transaction message is not older.

44. (Original) The medium of claim 39, wherein the determining (c) comprises:

(c1) determining that a target table is a child table of the referential integrity constraint; and

(c2) determining if a row operation of the transaction message is a row delete type; and

(c3) determining if the subsequent transaction message to a parent table is the row delete type, if the row operation of the transaction message is the row delete type.

45. (Original) The medium of claim 44, wherein the holding (d) comprises:

(d1) holding the subsequent transaction message to the parent table until the transaction message completes, if the row operation of the transaction message is not the row delete type;

(d2) holding the subsequent transaction message to the parent table, if the subsequent transaction message to the parent table is not the row delete type and the row operation of the transaction message is the row delete type; and

(d3) placing the subsequent transaction message to the parent table on the work queue to be applied in parallel with the transaction message, if the subsequent transaction message to the parent table is the row delete type and the row operation of the transaction message is the row delete type.

46. (Original) The medium of claim 45, wherein the placing (d3) further comprises:

(d3i) applying the subsequent transaction message to the parent table;

(d3ii) determining if a constraint violation results from the application of the subsequent transaction message; and

(d3iii) periodically retrying the application of the subsequent transaction message, if the constraint violation results.

47. (Original) The medium of claim 46, wherein the periodically retrying (d3iii) comprises:

(d3iiiA) comparing a next transaction message on the work queue with the transaction message to be retried;

(d3iiiB) determining if the next transaction message is older than the transaction message to be retried;

(d3iiiC) placing the transaction message to be retried back on the work queue, if the next transaction message is older; and

(d3iiiD) applying the transaction message to be retried, if the next transaction message is not older.

48. (Original) A computer readable medium with program instructions for preserving data constraints during parallel apply in asynchronous transaction replication in a database system, comprising:

(a) identifying a transaction message as a cascade delete;

(b) determining that a source of the transaction message is not a leaf table;

(c) placing each subscription for the transaction message onto a stack and placing row operations for each subscription into a reorder list, wherein the subscriptions are placed onto the stack in order of execution, wherein the row operations are placed into the reorder list in the order of execution; and

(d) adding the row operations for each subscription in the stack back to the transaction message, wherein the row operations are added in a reverse order of execution, wherein the subscriptions are added in the reverse order of execution.

49. (Original) The medium of claim 48, further comprising:

(e) sending the transaction message to be applied to a target table.

50. (Original) The medium of claim 48, further comprising:

(e) applying the transaction message at a target table.

51. (Original) A computer readable medium with program instructions for preserving data constraints during parallel apply in asynchronous transaction replication in a database system, comprising:

(a) receiving a message to perform an initial load of a target table;

(b) determining that the target table is a child table of referential integrity constraints;

(c) saving the referential integrity constraints for the target table;

(d) dropping the referential integrity constraints from the target table;

(e) loading the target table in parallel with a loading of a parent table of the referential integrity constraints;

(f) begin applying change data to the target table once loading is done;

(g) waiting for the parent table to finish loading, if the parent table has not yet finished loading; and

(h) adding the referential integrity constraints back into the target table.

52. (Original) The medium of claim 51, wherein the adding (h) comprises:

(h1) for each child referential integrity constraint for the target table, determining if a parent schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraints back into a child table of the target table, if the parent schema and table name of the referential constraint do not match the target table name in the subscription.

53. (Original) The medium of claim 52, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the parent schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the child table, if the state of the subscription is active or inactive.

54. (Original) The medium of claim 51, wherein the adding (h) comprises:

(h1) for each parent referential integrity constraint for the target table, determining if a child schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraint back into the target table, if the child schema and table name of the referential constraint do not match the target table name in the subscription.

55. (Original) The medium of claim 54, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the child schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the target table, if the state of the subscription is active or inactive.

56. (Original) A computer readable medium with program instructions for preserving data constraints during parallel apply in asynchronous transaction replication in a database system, comprising:

(a) receiving a message to perform an initial load of a target table;

(b) determining that the target table is a parent table of referential integrity constraints;

(c) saving the referential integrity constraints for a child table of the target table;

(d) dropping the referential integrity constraints from the child table;

(e) loading the target table in parallel with a loading of the child table;

(f) begin applying change data to the target table once loading is done;

(g) waiting for the child table to finish loading, if the child table has not yet finished loading; and

(h) adding the referential integrity constraints back into the child table.

57. (Original) The medium of claim 56, wherein the adding (h) comprises:

(h1) for each child referential integrity constraint for the target table, determining if a parent schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraints back into the child table, if the parent schema and table name of the referential constraint do not match the target table name in the subscription.

58. (Original) The medium of claim 57, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the parent schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the child table, if the state of the subscription is active or inactive.

59. (Original) The medium of claim 56, wherein the adding (h) comprises:

(h1) for each parent referential integrity constraint for the target table, determining if a child schema and table name of the referential constraint matches a target table name in a subscription; and

(h2) adding the referential integrity constraint back into the target table, if the child schema and table name of the referential constraint do not match the target table name in the subscription.

60. (Original) The medium of claim 59, wherein the adding (h) further comprises:

(h3) determining a state of the subscription, if the child schema and table name of the referential constraint matches the target table name in the subscription; and

(h4) adding the referential integrity constraints back into the target table, if the state of the subscription is active or inactive.

61. (Currently Amended) A system comprising:

~~a source node, wherein the source node sends a transaction message concerning a committed transaction completed at a source table copy to a target node to asynchronously replicate the transaction; and~~

~~the target node, wherein the target node comprises a receive queue, a browser thread, a work queue, and a target table copy;~~

a work queue;

a receive queue storing a transaction message, the transaction message comprising one or more row changes;

a browser thread in communication with the receive queue and the work queue, wherein the browser thread

~~receives and~~ examines the transaction message on the receive queue,

~~wherein the browser thread~~ determines ~~if~~ whether at least one row change in the transaction message ~~has~~ is affected by a data ~~constraints~~ constraint,

~~wherein if so~~ responsive to at least one row change in the transaction message being affected by a data constraint, ~~the browser thread~~ determines ~~if~~ whether there is a constraint violation between the at least one row change in the transaction message ~~has a constraint violation with~~ and a row change in at least one preceding non-completed transaction message, ~~and~~

~~wherein if so~~ responsive to there being a constraint violation between the at least one row change in the transaction message and the row change in the at least one preceding non-completed transaction message, ~~the browser thread~~ holds the transaction

message until application of the at least one preceding non-completed transaction message ~~completes~~ is completed, and

responsive to application of the at least one preceding non-completed transaction message being completed, places the transaction message on the work queue; and

one or more agent threads in communication with the work queue, wherein subsequent to placement of the transaction message on the work queue, the one or more agent threads apply the transaction message in parallel with one or more other transaction messages on the work queue.

62. (Cancelled)

63. (Cancelled)